

SN P and SNP Q systems

Eduardo Vázquez Fernández

Noviembre, 2023

- SN P systems ^a
- SNP Q systems ^b

^aCredits: Figures, tables and content of Spiking Neural P Systems were obtained from [1].

^bCredits: Figures, tables and content of Spiking Neural P Systems with Communication on Request were obtained from [2].

SN P systems [1]

Some Technical Prerequisites

- For an alphabet V , V^* is the free monoid generated by V with respect to the concatenation operation and the identity λ (the empty string).
- The set of all nonempty strings over V , that is, $V^* - \{\lambda\}$, is denoted by V^+ .
- When $V = \{a\}$ is a singleton, then we write simply a^* and a^+ instead of $\{a\}^*$, $\{a\}^+$.
- The length of a string $x \in V^*$ is denoted by $|x|$.

SN P systems [1]

Some Technical Prerequisites

- Regular languages are defined by means of regular expressions.
- Such expressions over a given alphabet V is constructed starting from λ and the symbols of V and using the operations of union, concatenation, and Kleene $+$, using parentheses when necessary for specifying the order of operations.

SN P systems [1]

Some Technical Prerequisites

Specifically,

- 1 λ and each $a \in V$ are regular expressions,
- 2 If $E1$ and $E2$ are regular expressions over V , then also $(E1) \cup (E2)$, $(E1)(E2)$, and $(E1)^+$ are regular expressions over V , and
- 3 Nothing else is a regular expression over V . With each expression E , we associate a language $L(E)$, defined in the following way:
 - 1 $L(\lambda) = \{\lambda\}$ and $L(a) = \{a\}$, for all $a \in V$.
 - 2 $L((E1) \cup (E2)) = L(E1) \cup L(E2)$, $L((E1)(E2)) = L(E1)L(E2)$, and $L((E1)^+) = L(E1)^+$, for all regular expressions $E1$ and $E2$ over V .

A language $L \subseteq V^*$ is said to be regular if there is a regular expression E over V such that $L(E) = L$.

SN P systems [1]

Spiking Neural P Systems

Specifically, we consider a spiking neural P system (in short, an SN P system), of degree $m \geq 1$, in the form

$$\Pi = (O, \sigma_1, \dots, \sigma_m, \text{syn}, i_0),$$

where:

- 1 $O = \{a\}$ is the singleton alphabet (a is called spike);
- 2 $\sigma_1, \dots, \sigma_m$ are neurons, of the form

$$\sigma_i = (n_i, R_i), \quad 1 \leq i \leq m,$$

SN P systems [1]

where:

- a) $n_i \geq 0$ is the initial number of spikes contained by the cell;
- b) R_i is a finite set of rules of the following two forms:
 - (1) $E/a^r \rightarrow a; t$, where E is a regular expression over O , $r \geq 1$, and $t \geq 0$;
 - (2) $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that $a^s \notin L(E)$ for any rule $E/a^r \rightarrow a; t$ of type (1) from R_i .
- ③ $\text{syn} \subseteq \{1, 2, \dots, m\} \times \{1, 2, \dots, m\}$ with $(i, i) \notin \text{syn}$ for $1 \leq i \leq m$ (synapses among cells);
- ④ $i_0 \in \{1, 2, \dots, m\}$ indicates the output neuron.

SN P systems [1]

$$E/a^r \rightarrow a; t$$

- The rules of type (1) are firing (we also say spiking) rules.
- If the contents of the neuron (the number of spikes present in it) is described by the regular expression E , r spikes are consumed and it produces a spike which will be sent to other neurons after t time units.
- A global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized.

SN P systems [1]

- There are two important actions which can take place in a step: **getting fired** and **spiking**.
- A neuron gets fired if the neuron contains n spikes such that $a^n \in L(E)$ and $n \geq r$.
- For instance, a rule $a(aa)^+ / a^3 \rightarrow a; 1$ can be applied to a neuron which contains seven copies of a , because $a^7 \in L(a(aa)^+) = \{a^{2n+1} \mid n \geq 1\}$, and then only four spikes remain in the neuron; now, the rule cannot be applied again – in general, the rule $a(aa)^+ / a^3 \rightarrow a; 1$ cannot be applied if the neuron contains any even number of spikes.

SN P systems [1]

- At the level of each neuron we work in a **sequential mode**, with at most one rule used in each step.
- At the level of the system we have a maximal **parallelism**, in the sense that in each step all neurons which can use a rule have to do it.

SN P systems [1]

- Now, about **spiking**. The use of a rule $E/a^r \rightarrow a; t$ in a step q means firing in step q and spiking in step $q + t$.
- That is, if $t = 0$, then the spike is produced immediately, in the same step when the rule is used.
If $t = 1$, then the spike will leave the neuron in the next step, and so on.
- In the interval between using the rule and releasing the spike, the neuron is assumed closed (in the **refractory period**), hence it cannot receive further spikes, and, of course, cannot fire again.

SN P systems [1]

- This means that if $t \geq 1$ and another neuron emits a spike in any moment $q, q + 1, \dots, q + t - 1$, then its spike will not pass to the neuron which has used the rule $E/a^r \rightarrow a; t$ in step q .
- In the moment when the spike is emitted, the neuron can receive new spikes.
- This means that if $t = 0$, then no restriction is imposed, the neuron can receive spikes in the same step when using the rule. Similarly, the neuron can receive spikes in moment t , in the case $t \geq 1$.

SN P systems [1]

- If a neuron σ_i spikes, its spike is replicated in all neurons σ_j such that $(i, j) \in \text{syn}$, and σ_j is open at that moment.
- If a neuron σ_i fires and all neurons σ_j such that $(i, j) \in \text{syn}$ are closed, then the spike of neuron σ_i is lost; the firing is allowed, it takes place, but it produces no spike.
- The rules of type 2 ($a^s \rightarrow \lambda$) are forgetting rules: s spikes are simply removed (“forgotten”) when applying. This rule is applied only if the neuron contains exactly s spikes.
- As defined above, the neurons can contain several rules. More precisely, it is allowed to have two spiking rules $E_1/ar_1 \rightarrow a; t_1$, $E_2/ar_2 \rightarrow a; t_2$ with $L(E_1) \cap L(E_2) \neq \emptyset$ (but not forgetting rules $a^s \rightarrow \lambda$ with $a^s \in L(E_i)$, $i = 1, 2$). The rules are used in the non-deterministic manner, in a maximally parallel way at the level of the system.

SN P systems [1]

- A spike emitted by a neuron i will pass immediately to all neurons j such that $(i, j) \in \text{syn}$ and are open.
- The transmission of a spike takes no time, the spikes are available in the receiving neurons already in the next step.
- The **initial configuration** is defined by the number of initial spikes n_1, \dots, n_m .

SN P systems [1]

- **A transition** is defined as the pass from a configuration of the system to another configuration. For two configurations C_1, C_2 of Π , we denote by $C_1 \Rightarrow C_2$ the fact that there is a direct transition from C_1 to C_2 in Π .
- The **transitive closure** of the relation \Rightarrow is denoted by \Rightarrow^* .
- A sequence of transitions, starting in the initial configuration, is called a **computation**.

SN P systems [1]

- With a computation we can associate **several results**.
- One possibility is count the **number of spikes** present in the output neuron in the halting configuration.
- Another possibility, we do not care whether or not the computation halts, but we only request that **the output neuron spikes exactly twice** during the computation. Then, **the number of steps elapsed between the two spikes** is the number computed by the system along that computation.
- We denote by $N_2(\Pi)$ the set of numbers computed in this way by a system Π , with the subscript 2 reminding of the way the result of a computation is defined.

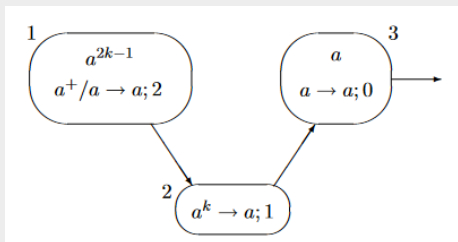
SN P systems [1]

Examples

- Each neuron is represented by a “membrane” (a circle or an oval), marked with a label and having inside both the current number of spikes (in the form a^n for n spikes present in a neuron) and the evolution rules.
- The synapses linking the neurons are represented by arrows.
- The output neuron will be identified by its label (i_0) and a short arrow which exits from it, pointing to the environment.
- **Convention:** if a spiking rule is of the form $E/a^r \rightarrow a; t$, with $L(E) = \{a^r\}$ (this means that such a rule is applied when the neuron contains exactly r spikes – and all these spikes are consumed), then we will write this rule in the simpler form $a^r \rightarrow a; t$.

SN P systems [1]

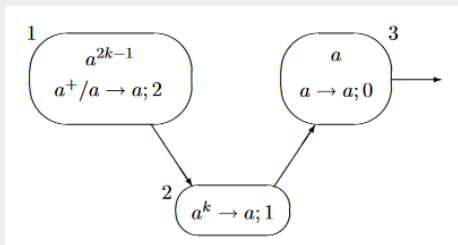
A simple example of an SN P system



In the system Π_1 , we have three neurons, with labels 1, 2, 3; neuron 3 is the output one. In the initial configuration, we have spikes in neurons 1 and 3, and these neurons fire already in the first step. The spike of neuron 3 exits the system, so the number of steps from now until the next spiking of neuron 3 is the number computed by the system. After firing, neuron 3 remains empty, so it cannot spike again before receiving a new spike.

SN P systems [1]

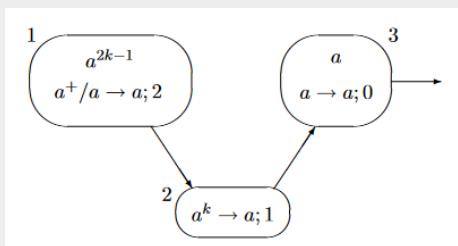
A simple example of an SN P system



In turn, neuron 2 cannot fire until collecting exactly k spikes. After firing, neuron 1 will be closed/blocked for the next two steps; in the third step it will release its spike, sending it to neuron 2, and in step 3 will fire again. Thus, neuron 1 fires in every third step, consuming one of the spikes: any number $n \geq 1$ of spikes is “covered” by the regular expression a^+ .

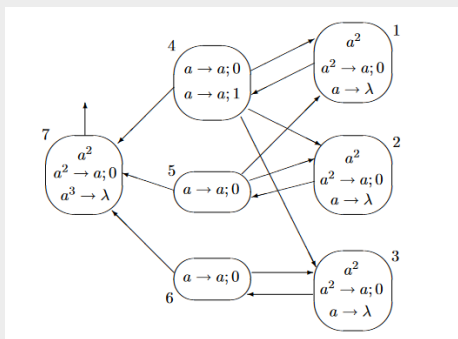
SN P systems [1]

A simple example of an SN P system



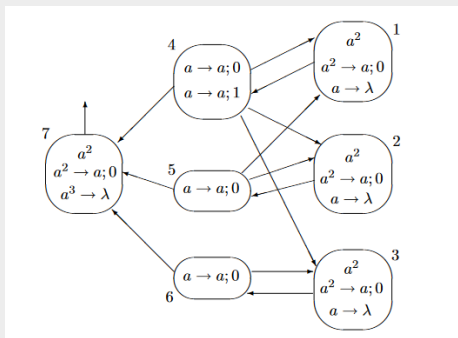
In the step $3k$, neuron 2 will receive the k th spike emitted by neuron 1, hence in the next moment, $3k + 1$, it will fire. The delay between firing and spiking is of one time unit for neuron 2, hence its spike will reach neuron 3 in step $3k + 2$, meaning that neuron 3 spikes again in step $3k + 3$. Therefore, **the computed number is** $(3k + 3) - 1 = 3k + 2$.

SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

In the beginning, only neurons 1, 2, 3, and 7 (output neuron) contain spikes, hence they fire in the first step – and spike immediately. The output neuron spikes, hence we have to count the number of steps until the next spike, to define the result of the computation.

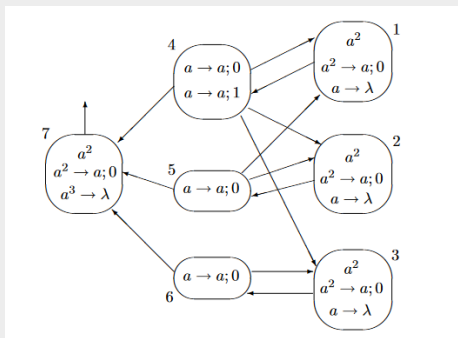
SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

Note that in the first step we cannot use the forgetting rule $a \rightarrow \lambda$ in neurons 1, 2, 3, because we have more than one spike present in each neuron.

The spikes of neurons 1, 2, 3 will pass to neurons 4, 5, 6.

SN P systems [1]

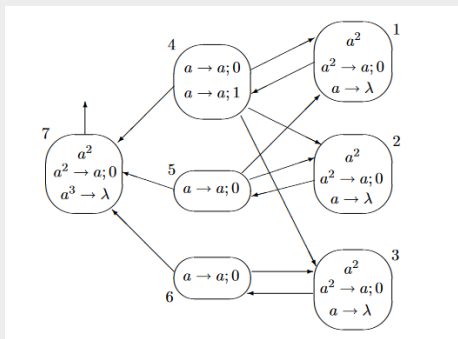
An SN P system generating all even natural numbers (Π_2)

In step 2, neurons 1, 2, 3 contain no spike inside, hence will not fire, but neurons 4, 5, 6 fire.

Neurons 5, 6 have only one rule, but neuron 4 behaves non-deterministically, choosing between the rules.

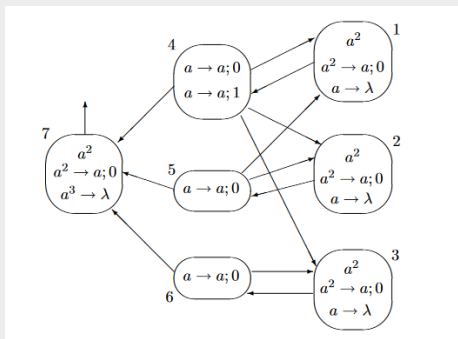
SN P systems [1]

An SN P system generating all even natural numbers (Π_2)



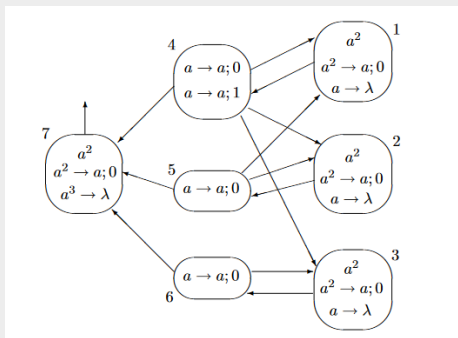
Assume that for $m \geq 0$ steps we use here the first rule. This means that three spikes are sent to neuron 7, while each of neurons 1, 2, 3 receives two spikes.

SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

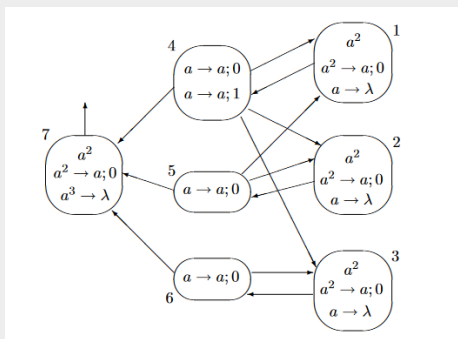
In step 3, neurons 4, 5, 6 cannot fire, but all neurons 1, 2, 3 fire again. After receiving the three spikes, neuron 7 uses its forgetting rule and gets empty again. These steps can be repeated arbitrarily many times.

SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

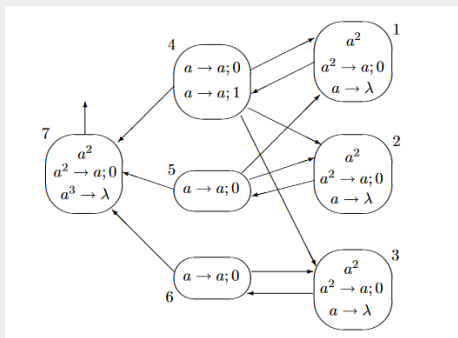
In order to fire again neuron 7, we have to use sometimes the rule $a \rightarrow a; 1$ of neuron 4. Assume that this happens in step t (this happens in $t = 2m + 2$). This means that at step t only neurons 5, 6 emit their spikes.

SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

Each of neurons 1, 2, 3 receives only one spike – and forgets it in the next step, $t + 1$. Neuron 7 receives two spikes, and fires again, thus sending the second spike to the environment. This happens in moment $t + 1 = 2m + 2 + 1$, hence **the computed number is $2m + 2$** ($m \geq 0$).

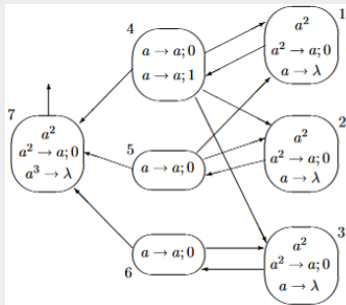
SN P systems [1]

An SN P system generating all even natural numbers (Π_2)

The spike of neuron 4 will reach neurons 1, 2, 3, and 7 in step $t + 1$, hence it can be used only in step $t + 2$; in step $t + 2$ neurons 1, 2, 3 forget their spikes and the computation halts. The spike from neuron 7 remains unused, there is no rule for it.

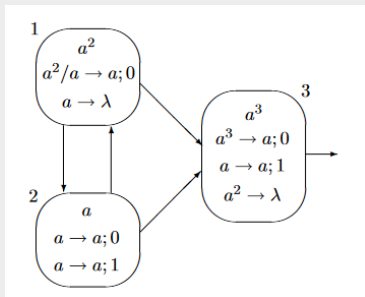
SN P systems [1]

The number 4 is calculated



Step	0	1	2	3	4	5	6
Neuron 1		$a^2 \rightarrow a; 0$	—	$a^2 \rightarrow a; 0$	—	$a \rightarrow \lambda$	$a \rightarrow$
Neuron 2		—	$a_4 a_5$	—	a_5	a_4	—
Neuron 3		—	$a_4 a_6$	—	a_6	a_4	—
Neuron 4		a_1	—	a_1	—	—	—
Neuron 5		—	a_2	—	a_2	—	—
Neuron 6		—	a_3	—	a_3	—	—
Neuron 7		$a^2 \rightarrow a; 0 !$	—	$a^3 \rightarrow \lambda$	—	$a^2 \rightarrow a; 0 !$	—
	aa	—	$a_4 a_5 a_6$	—	$a_5 a_6$	a_4	a

SN P systems [1]

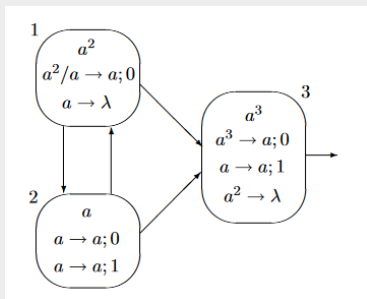
Generating all natural numbers greater than 1 (Π_3)

$$\Pi_3 = (\{a\}, \sigma_1, \sigma_2, \sigma_s, \mathbf{syn}, 3), \sigma_1 = (2, \{a^2/a \rightarrow a; 0, a \rightarrow \lambda\}),$$

$$\sigma_2 = (1, \{a \rightarrow a; 0, a \rightarrow a; 1\}), \sigma_3 = (3, \{a^3 \rightarrow a; 0, a \rightarrow a; 1, a^2 \rightarrow \lambda\}),$$

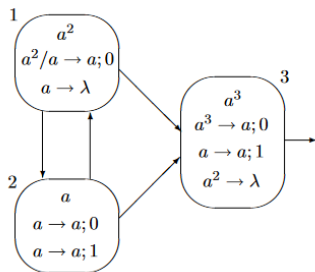
$$\mathbf{syn} = \{(1, 2), (2, 1), (1, 3), (2, 3)\}.$$

SN P systems [1]

Generating all natural numbers greater than 1 (Π_3)

Both neurons 1 and 2 send a spike to the output neuron, 3; these two spikes are forgotten in the next step. Neurons 1 and 2 also exchange their spikes; thus, as long as neuron 2 uses the rule $a \rightarrow a; 0$, the first neuron receives one spike, thus completing the needed two spikes for firing again.

SN P systems [1]

Generating all natural numbers greater than 1 (Π_3)

SNQ P systems [2]

Introduction

- SN P systems and their variants are communicating on command: the initiative for communication belongs to the **emitting neuron**.
- Taking inspiration from the area of Parallel-Cooperating Grammar Systems, it is natural to consider also the reverse case: communication on request. The spikes should be moved from a neuron to another one when the **receiving neuron** requests that.

Examples of request-response communication

- Browsing a web page is an example of request-response communication.
- Request-response can be seen as a telephone call, in which someone is called and they answer the call.

SNQ P systems [2]

Introduction

- Now, we will see the **SN P Systems with Communications by Request** (SNQ P systems).
- They have only rules for requesting spikes from the neighboring neurons, the action being again dependent on the contents of the neuron.
- Basically, the rules are of the form $E/Q(a^{n_1}, j_1) \dots (a^{n_m}, j_m)$, with the meaning that, if the neuron where this rule resides (say, neuron i) has a number of spikes described by the regular expression E , then it asks (this is the meaning of Q) n_1 spikes from neuron j_1 , n_2 spikes from neuron j_2 , and so on.
- If the neurons j_1, \dots, j_m cannot satisfy the requests (they contain fewer spikes than requested), then the rule cannot be applied.

SNQ P systems [2]

Introduction

- Also queries of the form (a^∞, j) can be formulated, with the meaning that all spikes from neuron j are requested, no matter how many they are, maybe none.
- When several neurons request simultaneously spikes from the same neuron, the queries should be identical, and the requested spikes are replicated.
- Important feature: no spike is consumed, they are only moved from a neuron to another one. The only way to **increase** the number of spikes in the systems is by replicating the spikes in neurons which receive multiple queries.

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

Definition (SNQ P Systems)

A spiking neural P system with communication by request (shortly, SNQ P system) is a construct

$$\Pi = (O, \sigma_1, \dots, \sigma_m, a_{i_0}, \text{out}),$$

where

- 1 $O = \{a_1, a_2, \dots, a_k\}$ is an alphabet (a_i is a type of spikes), where $k \geq 1$ is the number of types of spikes,
- 2 $\sigma_1, \dots, \sigma_m$ are neurons of the form $\sigma_i = (u_i, R_i)$, $1 \leq i \leq m$, m is the number of neurons, where:
 - (a) u_i is a multiset over the alphabet O ,

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

$$\Pi = (O, \sigma_1, \dots, \sigma_m, a_{i_0}, \text{out}),$$

- (b) R_i is a finite set of rules of the form E/Qw , where E is a regular expression over O , and w is a finite set of queries of the forms (a_s^p, j) and (a_s^∞, j) , $1 \leq s \leq k$, $p \geq 0$, $1 \leq j \leq m$,
- 3 a_{i_0} , $1 \leq i_0 \leq k$, is the type of output spikes, and $\text{out} \in \{1, 2, \dots, m\}$ indicates the output neuron, which is used to store the computation result.

The meaning of a query (a_s^p, j) is that neuron σ_i requests p copies of a_s from neuron σ_j , while (a_s^∞, j) means that all spikes of type a_s from σ_j , no matter how many they are, are requested by σ_i .

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

It must be noted that the set of synapses has not been specified in the definition since the synapses are implicitly defined by the rules.

A configuration C_t at an instant t of an SNQ P system

$\Pi = (O, \sigma_1, \dots, \sigma_m, a_{i_0}, \text{out})$, where $\sigma_i = (u_i, R_i)$ and $u_i = a_1^{n(i,1)} \dots a_k^{n(i,k)}$, $1 \leq i \leq m$, is described by the number of spikes of each type present in each neuron at the beginning of the computation, that is,

$$C_t = ((n(1,1) \dots n(1,k)), \dots, (n(m,1) \dots n(m,k))).$$

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

A rule E/Qw , with a finite set of queries w of the form (a_s^p, j) , in neuron σ_i is applicable to a configuration C_t at time t if the following holds:

- The contents of neuron σ_i belongs to the language generated by E .
- All queries formulated in w are satisfied, that is, if (a_s^p, j) is a query in w then neuron σ_j has at least p spikes (all queries (a_s^∞, j) in w are always satisfiable because all spikes from σ_j are requested for neuron σ_i , no matter how many they are, maybe none).

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

- There could exist conflicting queries between two rules $r_1 \equiv E_1/Qw_1$ and $r_2 \equiv E_2/Qw_2$ associated with neurons σ_{i_1} and σ_{i_2} verifying conditions (1), (2), and different numbers of occurrences of the same spike as of neuron σ_j are requested by σ_{i_1} and σ_{i_2} . In this case, one of the rules r_1, r_2 , nondeterministically chosen, can be used.
- In this case, one of the rules r_1, r_2 , nondeterministically chosen, can be used.

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

A computation step consists of the following sub-steps:

- **Sub-step 1.** In each neuron, we choose a rule to apply, and check its applicability. This means checking three conditions: (i) that the regular expression in the rule corresponds to the contents of the neuron, (ii) that the queries in the rule can be satisfied by the indicated neurons, and (iii) that there are no conflicting queries among the selected rules.

The set of selected applicable rules should be maximal (each neuron which can evolve, should do it).

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

A computation step consists of the following sub-steps:

- **Sub-step 2.** The requested spikes are removed from the neurons where they were present. For each neuron we have three cases: (i) no spike a_s was requested by any other neuron (and then the existing number of spikes a_s remains unchanged), (ii) all spikes of some kind a_s were requested, by at least one other neuron (and then no spike of this kind remains here), or (iii) p spikes of type a_s are requested, by at least one other neuron (and then p is deduced from the number of copies of a_s present in the neuron).

SNQ P systems [2]

Definition of SN P Systems with Communication by Request

- The previous sub-steps together form a step, which lasts for one time unit.
- After a computation step as illustrated above, the system passes to a new configuration. A sequence of such transitions from a configuration to another one, starting from the initial configuration, is called a **computation**.
- A **computation halts** if it reaches a configuration where no rule can be applied. The result of a halting computation is the number of copies of spike a_{i0} present in neuron σ_{out} in the halting configuration.

SNQ P systems [2]

Differences between standard SN P systems and the proposed SNQ P systems

It must be noted that there exist several important **differences** of SNQ P systems in comparison with usual SN P systems:

- We use several types spikes.
- There is no interaction with the environment, no spike is sent out (no spike train is defined here).
- There is no other way to increase the number of spikes than the **replication** in the case of multiple queries from the same neuron (this corresponds to the case when a neuron in a usual SN P system sends spikes to several neurons to which it has synapses).

SNQ P systems [2]

Examples

- Now, we will study the computational power of SNQ P systems with small number of neurons and of types of spikes.
- Let us denote by $N(\Pi)$ the set of numbers generated by Π .
- Let us also denote by $NSN_kP_m(Q)$ the family of sets $N(\Pi)$ generated by SNQ P systems using at most k types of spikes and at most m neurons.
- When the numbers k or m can be arbitrary, the corresponding parameter is replaced with $*$.
- The two parameters k and m induce a double hierarchy of families of sets of numbers:

$$NSN_kP_m(Q) \subseteq NSN_{k+1}P_{m+1} \text{ for all } k \geq 1, m \geq 1.$$

SNQ P systems [2]

Examples

Systems with only **one neuron** cannot apply any rule, hence they **only generate singleton sets**.

Lemma

*Lemma 1. $NSN_kP1 = NSN_*P1 = SING$, $k \geq 1$, where $SING$ denotes the family of singleton sets.*

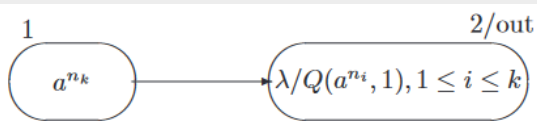
SNQ P systems [2]

Examples

Systems with two neurons have also a rather limited power.

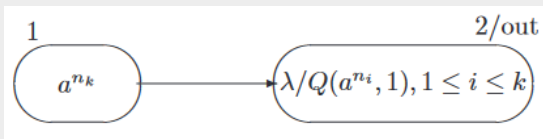
Lemma

*Lemma 2. $NSN_kP2 = NSN_*P2 = FIN$, $k \geq 1$, where FIN denotes the family of finite sets of numbers.*



SNQ P systems [2]

Examples



Proof

- $NSN_*P_2 \subseteq FIN$: In systems with two neurons, **spikes cannot be replicated**, hence the initial number of spikes cannot be increased.
- $FIN \subseteq NSN_1P_2$: Initially, neuron σ_1 contains n_k spikes, while σ_2 is empty, corresponding to the empty string λ . Each computation takes only one step, neuron σ_2 non-deterministically chooses rule $\lambda/Q(a^{n_i}, 1)$ to apply, requesting n_i spikes from neuron σ_1 . In this way, **the number n_i is generated** ($1 \leq i \leq k$).

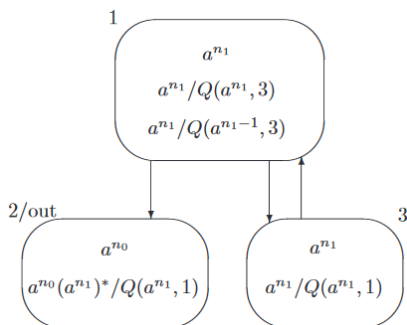
SNQ P systems [2]

Examples

The passage from 2 to 3 neurons increase of the generative power permitting the increase of number of spikes (replication of spikes).

Lemma

Lemma 3. The family $NSN_1P_3(Q)$ contains any arithmetical progression.



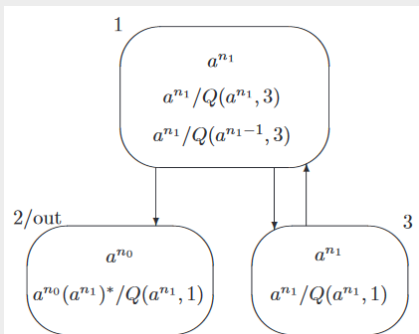
SNQ P systems [2]

Examples

Proof

Let us take an arithmetical progression $L = \{n_0 + i \cdot n_1 | i \geq 1\}$, $n_0 \geq 0$ and $n_1 \geq 1$. Formally the SNQ P system Π is:

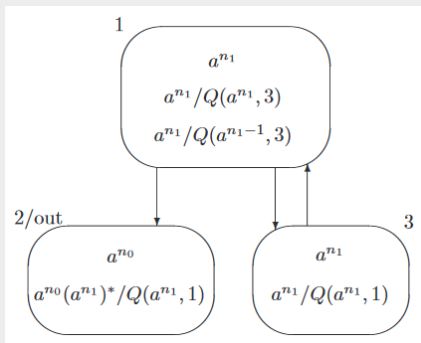
$$\Pi = (\{a\}, \sigma_1, \sigma_2, \sigma_3, a, 2), \sigma_1 = (n_1, \{a_{n_1}/Q(a_{n_1}, 3), a_{n_1}/Q(a_{n_1-1}, 3)\}), \\ \sigma_2 = (n_0, \{a_{n_0}(a_{n_1})^*/Q(a_{n_1}, 1)\}), \sigma_3 = (n_1, \{a_{n_1}/Q(a_{n_1}, 1)\}).$$



SNQ P systems [2]

Examples

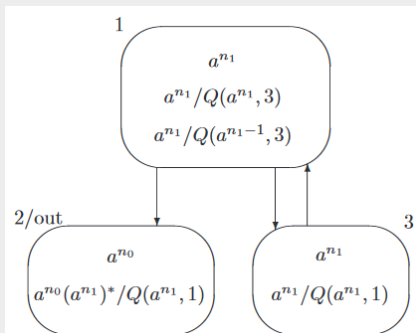
In each step, neurons σ_1 and σ_3 repeatedly exchange n_1 spikes, while neuron σ_2 also requests n_1 spikes from neuron σ_1 (hence the n_1 spikes of neuron σ_1 are duplicated).



SNQ P systems [2]

Examples

The computation can stop at any moment, by using the second rule of neuron σ_1 : neuron σ_1 brings only $n_1 - 1$ spikes inside (hence the query of neuron σ_2 cannot be satisfied) and neuron σ_3 remains with one spike inside, which, together with the n_1 spikes brought from neuron σ_1 , do not allow the use of the rule in neuron σ_3 . Clearly, $N(\Pi) = L$.

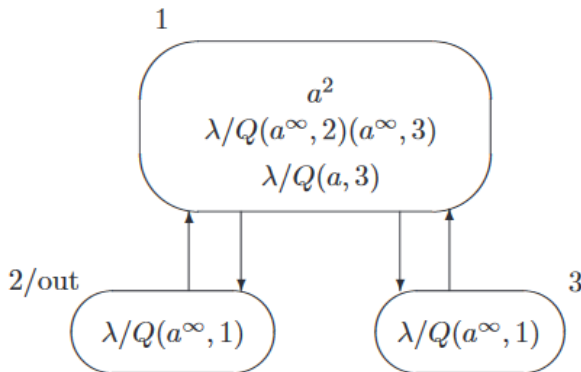


SNQ P systems [2]

Examples

Lemma

Lemma 4. The family $NSN_1P_3(Q)$ contains non-semilinear sets of numbers.

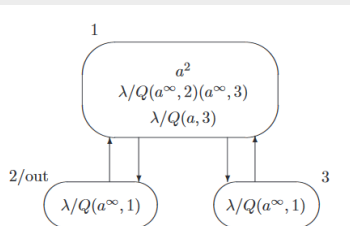


SNQ P systems [2]

Examples

Proof

$\sigma_1, \sigma_2, \sigma_3$ can use a rule only if they are empty. This is the case in the beginning with neurons σ_2 and σ_3 , hence they request all spikes of neuron σ_1 . Now neuron σ_1 can request back the spikes from neurons σ_2, σ_3 , getting 4 spikes inside. As long as the neurons use their rules asking for all spikes of the partner neurons, the number of spikes present in neuron σ_1 is doubled (this also happens with the contents of neurons σ_2 and σ_3).

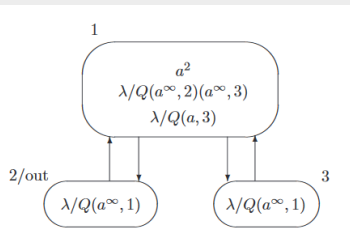


SNQ P systems [2]

Examples

Proof

At some step, $\lambda/Q(a, 3)$ is used in σ_1 , simultaneously with σ_2 and σ_3 requesting the spikes of σ_1 . The computation halts because all neurons have at least one spike inside. The number of spikes in σ_1 is doubled after each move of the contents of neurons σ_2 and σ_3 to σ_1 . After σ_2 requests all spikes from σ_1 , the number of spikes in the output neuron σ_2 is a power of 2, $N(\Pi) = \{2^n | n \geq 1\}$.



SNQ P systems [2]

Universality of SNQ P systems

- Therefore, the increase of the number of neurons from 1 to 2 and to 3 increase of the computing power of SNQ P systems.
- What happens to the next levels of the hierarchy?

$$NSN_k P_m(Q) \subseteq NSN_k P_{m+1}.$$

SNQ P systems [2]

Universality of SNQ P systems

Theorem

Theorem 1 (Universality of SNQ P Systems). SNQ P Systems are computationally universal: $NRE = NSN_2P_(Q)$.*

- The proof will use the representation of NRE (the family of sets of natural numbers computed by Turing machines) by means of **register machines**.
- The set of all numbers computed by register machine M is denoted by $N(M)$. It is known that register machines compute all sets of numbers which are Turing computable, hence they represent NRE .

SNQ P systems [2]

Universality of SNQ P systems

Proof

In order to prove the theorem, we only prove the inclusion \subseteq ; the opposite one can be obtained through a straightforward construction of a Turing machine simulating an SNQ P system.

A register machine $M = (n, H, l_1, l_h, I)$, where n is the number of registers, H is the set of instruction labels, l_1 is the start label, l_h is the halt label, and I is the set of instructions. The instructions are of the following forms:

- $l_i : (ADD(r), l_j, l_k)$ (add 1 to register r and then go to one of the instructions with labels l_j, l_k),
- $l_i : (SUB(r), l_j, l_k)$ (if register r is nonempty, then subtract 1 from it and go to the instruction with label l_j , otherwise go to the instruction with label l_k),
- $l_h : \text{HALT}$ (the halt instruction).

SNQ P systems [2]

Universality of SNQ P systems

- A register machine M starts with all registers with zero.
- It applies the instruction with label l_1 , and proceeds to apply instructions as indicated by labels.
- If the machine reaches the halt instruction, then the number n stored in the first register is said to be computed by M .
- The set of all numbers computed by M is denoted by $N(M)$. It is known that register machines compute all sets of numbers which are Turing computable, hence they characterize NRE .

SNQ P systems [2]

Universality of SNQ P systems

- Starting from a register machine $M = (n, H, l_0, l_h, I)$, we construct an SNQ P system Π with two types of spikes, which we denote by a and b , hence $O = \{a, b\}$.
- We associate one neuron σ_l with each label $l \in H$.
- We associate a second neuron, $\sigma_{l_i}^i$ with each instruction of M of the form $l_i : (SUB(r), l_j, l_k)$. We also consider the neurons σ_i , $1 \leq i \leq 5$.

SNQ P systems [2]

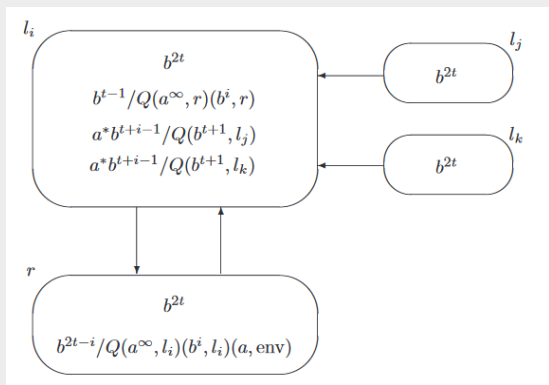
Universality of SNQ P systems

- If the value of a register r is m , then σ_r contains m spikes a . The output neuron is σ_1 (associated with register 1).
- Let us assume that H contains t elements, l_1, l_2, \dots, l_t . Initially, each neuron contains $2t$ copies of b and no copy of a , with the exception of neurons σ_{c_i} , $1 \leq i \leq 5$, which contain the spikes a, b, b, b, b , respectively.

SNQ P systems [2]

Universality of SNQ P systems

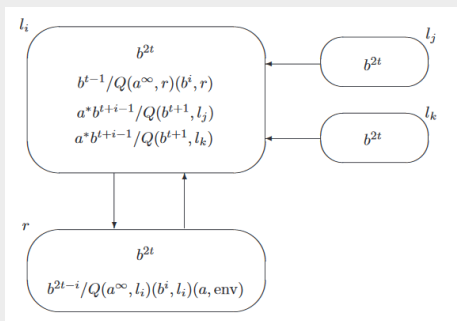
ADD module: For each instruction $l_i : (ADD(r), l_j, l_k)$ of M , we construct the next module.



SNQ P systems [2]

Universality of SNQ P systems

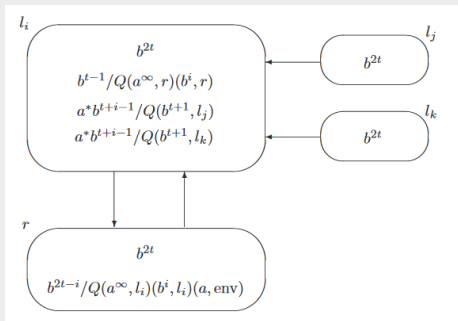
- Such a module is activated when $t + 1$ spikes b from neuron l_i are requested by another neuron, then we have to start with only $t - 1$ spikes b .
- The neuron (with label) l_i becomes active and it requests from neuron r all copies of a as well as i copies of b .



SNQ P systems [2]

Universality of SNQ P systems

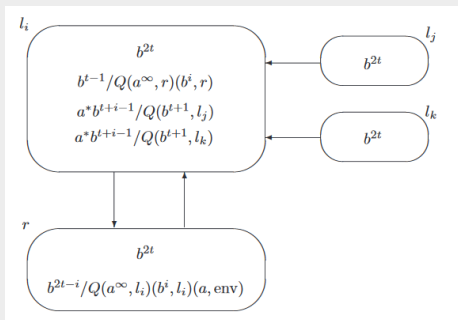
In the next step, both neurons l_i and r can apply a rule. In this way, the previous contents of neuron r returns to neuron r , at the same time neuron r requests one copy of a from the environment, which corresponds to the fact that the register was increased by 1.



SNQ P systems [2]

Universality of SNQ P systems

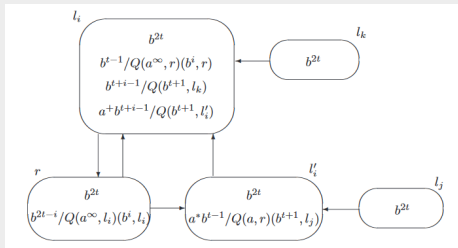
Simultaneously, neuron l_i uses one of the rules $a^*b^{t+i-1}/Q(b^{t+1}, l_j)$, $a^*b^{t+i-1}/Q(b^{t+1}, l_k)$, nondeterministically chosen, which means that one of the neurons l_j, l_k is activated, while l_i ends with $2t$ copies of b inside, as it was the case at the beginning.



SNQ P systems [2]

Universality of SNQ P systems

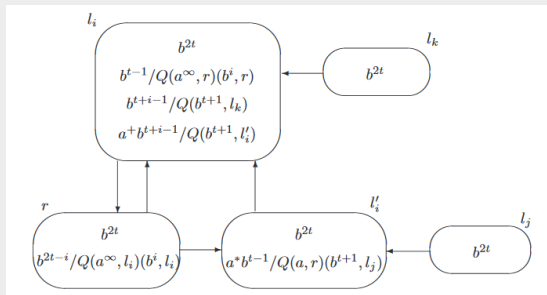
SUB module: For each instruction $l_i : (SUB(r), l_j, l_k)$ of M , we construct the next module:



SNQ P systems [2]

Universality of SNQ P systems

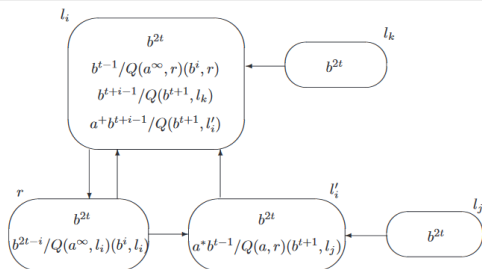
When l_i “loses” $t + 1$ spikes b , it becomes active and can absorb all spikes a and i spikes b from neuron r . In the next step, both neurons r and l_i can use one rule. If there is no spike a present (corresponding to the fact that register r was empty), then neuron l_i has to use the rule $b^{t+i-1}/Q(b^{t+1}, l_k)$, and neuron l_k is activated. In parallel, neuron r returns to its previous contents (no subtraction was made).



SNQ P systems [2]

Universality of SNQ P systems



- If there is at least one copy of spike a present, **the subtraction is performed** by activating first neuron l'_i (in parallel, neuron r returns to its previous contents).
- Neuron l'_i decreases by one the contents of neuron r and activates neuron l_j .
- The number of copies of spike a in neuron 1 is the result of computation, hence $N(M) = N(\Pi)$.



Some Applications

- Implementation of RSA cryptographic algorithm using SN P systems based on HP/LP neurons.
- Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources.
- A Cluster Splitting Technique by Hopfield Networks and P Systems on Simplices
- A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection.

References

-  Mihai Ionescu, Gheorghe Păun, and Takashi Yokomori.
[1] Spiking neural p systems.
Fundamenta informaticae, 71(2-3):279–308, 2006.
-  Linqiang Pan, Gheorghe Paun, Gexiang Zhang, and Ferrante Neri.
[2] Spiking neural p systems with communication on request.
International Journal of Neural Systems, 27, 08 2017.